Analyzátor CAN sběrnice s rozhraním Ethernet

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Martin Košarko**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Kybernetika a měření, blok KM1 - Řídicí technika

Název tématu: **Analyzátor CAN komunikace s rozhraním Ethernet**

Pokyny pro vypracování:

1. Seznamte se s protokoly CAN 2.0 a CANopen (dle CiA DS301). Seznamte se s HW platformou firmy POLL a souvisejícími SW prostředky.
2. Na této plátformě navrhněte analyzátor CAN/CANopen, který bude obsahovat především tyto funkčnosti:
- automatický záznam sledovaných zpráv na paměťovou kartu, který bude probíhat autonomně bez připojení PC,
- možnost odesílání uživatelských zpráv na sběrnici CAN,
- vizualizaci zpráv na sběrnici CAN včetně volitelné interpretace dle CAN nebo CANopen.
3. Uvedené funkčnosti implementujte a otestujte v ostrém provozu.
4. Zpracujte programátorskou a uživatelskou dokumentaci.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Pavel Burget, Ph.D.

Platnost zadání: do konce letního semestru 2011/2012

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Boris Šimák, CSc.
děkan

V Praze dne 22. 2. 2011

Poděkování

Na tomto místě bych rád poděkoval vedoucímu diplomové práce Ing. Pavlu Burgetovi, Ph.D., zejména za trpělivost, ochotu a vstřícnost při mém vedení. Dále bych chtěl poděkovat své rodině a kolegům, kteří se podíleli na vytvoření příjemného prostředí, v němž tato práce mohla vzniknout.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, dne …… ……

podpis

Abstrakt

Tato práce se zabývá návrhem a implementací SW pro diagnostické zařízení sběrnice CAN na HW platformě firmy POLL. Obsahuje rozbor používaných webových technologií, návrh komunikačního protokolu přes sběrnici ethernet a návrh jednotlivých modulů realizujících zvolené funkčnosti. Výsledkem je zhodnocení daného návrhu a funkční SW.

Klíčová slova: CAN, záznamové zařízení, ethernet

Summary

This thesis deals with problem of CAN messages data-logging. Device implementing this functionality is controlled via ethernet bus. This work comprises summary of basic needs for such unit, their design and implementation. The result of this work is functional SW used on device designed by POLL company.

Keywords: CAN, data-logger, ethernet

Obsah

1 ÚVOD 1

2 HW 2

2.1 SOUČASNÉ PRODUKTY 2
2.1.1 National Instruments 2
2.1.2 Kvaser 3
2.1.3 ADFweb 3
2.1.4 Další produkty 3
2.1.5 Závěr 3
2.2 POPIS HW PLATFORMY POLL 3

3 ANALÝZA ZADÁNÍ 6

3.1 WEBOVÉ TECHNOLOGIE 7
3.1.1 Internetové aplikace založené na Flash Player 7
3.1.2 AJAX 7
3.1.3 Websocket 8
3.1.4 Server-Sent Events 9
3.1.5 JSON 9
3.1.6 Silverlight 10
3.1.7 Závěr 11
3.2 CAN 12
3.2.1 CANopen 12
3.2.1.1 Rejstřík objektů CANopen 12
3.2.1.2 Objekt SYNC 13
3.2.1.3 Time Stamp (TIME) 13
3.2.1.4 PDO protokol 13
3.2.1.5 SDO protokol 14
3.2.1.6 NMT protokol 14
3.2.2 Nastavení CAN sběrnice 14
3.2.3 Vysílání vlastních zpráv po CAN 15
3.2.4 Příchozí zprávy po sběrnici CAN 17
3.3 PRÁCE SE SOUBORY 17
3.3.1 Záznamové médium 17
3.3.2 FileSystem v programu 19

4 DESIGN 20

4.1 SERVER 20
4.1.1 http_state 20
4.2 MODUL KOMUNIKACE 21
4.2.1 Komunikační protokol 21

4.2.1.1 getCANM 22
4.2.1.2 createDir 23
4.2.1.3 deleteDir 23
4.2.1.4 createFile 23
4.2.1.5 deleteFile 24
4.2.1.6 showDirTree 24
4.2.1.7 setCANBus 25
4.2.1.8 setCANM 25
4.2.1.9 stopCANM 26
4.2.1.10 deleteCANM 26
4.2.1.11 saveCANM 26
4.2.1.12 readCANBusSet 26
4.2.1.13 readCANMset 27
4.2.1.14 Chybové hlášky 28
4.3 MODUL CAN 28

5 IMPLEMENTACE 29

5.1 POUŽITÉ TECHNOLOGIE 29
5.2 FREERTOS 29
5.3 MODUL SOUBOROVÉHO SYSTÉMU 30
5.4 METODA MAIN 31
5.5 FILE MANAGER 31
5.6 MODUL CAN 32
5.6.1 canDriver 32
5.6.1.1 getMailBox 32
5.6.1.2 configureBus 33
5.6.1.3 configureMailBoxSend 33
5.6.1.4 configureMailBoxRec 33
5.6.1.5 freeMailbox 33
5.6.2 mailBox 33
5.7 KLIENT 34
5.7.1 Obecně 34
5.7.2 Komunikační modul v JS 34
5.7.3 User-Interface 35
5.7.3.1 Aktuální provoz 35
5.7.3.2 Nastavení sběrnic 35
5.7.3.3 Nastavení schránek 36
5.7.3.4 Souborový systém 38
5.7.3.5 Chybové hlášky 39

6 ZÁVĚR 40

Seznam obrázků

Obr. 2.1 DPS 5
Obr. 2.2 Týl zařízení 5
Obr. 2.3 Čelo zařízení 5
Obr. 3.1 Stavový automat schránek CAN 16
Obr. 4.1 HTTP_state 20
Obr. 4.2 Stromová struktura 25
Obr. 5.1 Modul souborového systému 30
Obr. 5.2 Třída canDriver 32
Obr. 5.3 Nastavení sběrnic 36
Obr. 5.4 Nastavení zpráv CAN 37
Obr. 5.5 Ovládání souborů 38
Obr. 5.6 Zobrazení chybové hlášky 39

Seznam tabulek

Tab. 3.1 Pro a proti webových technologií 11
Tab. 3.2 Rejstřík objektů CANopen 13
Tab. 4.1 Seznam zpráv 22
Tab. 4.2 Formát CAN zprávy 23
Tab. 4.3 Formát nastavení sběrnice 27
Tab. 4.4 Formát nastavení CAN schránek 27
Tab. 4.5 Chybové kódy 28

1 Úvod

V dnešní době je téměř nemyslitelné narazit na zařízení, které by se obešlo bez mikroprocesoru, ať už jde o velké výrobní haly, řídící jednotky v automobilech, nebo otvírání dveří v tramvaji. Samotné zařízení však nestačí, pokud není s ostatním okolím propojeno nějakou formou komunikace, ať už přes sběrnice USB, CAN či ethernet. Zejména poslední zmíněná sběrnice se v rámci několik posledních let v průmyslovém prostředí rozmáhá.

V technologických systémech a při použití různých sběrnic však vlivem okolního prostředí může docházet k EMC rušení, které například způsobuje problémy s komunikací mezi jednotlivými zařízeními. Je tedy důležité mít možnost sledovat provoz na jednotlivých datových sběrnicích a být schopen eventuální výpadky a poruchy diagnostikovat a dát je do souvislostí s ostatními jevy.

Cílem této práce je navrhnout SW modul pro zařízení, který bude schopný sledovat a v reálném čase zobrazovat dění na sběrnici CAN. Další funkčností je dlouhodobý záznam tohoto dění na paměťové médium. V současné době již existuje několik produktů realizujích záznam zpráv na sběrnici CAN, které se však z různých důvodů neukázaly jako vhodné pro zadavatele, jejich částečný výčet je uveden v kapitole 2. Proto bylo přistoupeno k vývoji na HW zadavatele, ten je popsán ve stejné kapitole.

V kapitole 3 se věnuji rozboru současných komunikačních technologií, popisu protokolu CAN a CANopen. Další částí této kapitoly je rozbor požadavků na ovládání aplikace. Kapitola 4 následně pojednává o navrženém komunikačním protokolu. V poslední kapitole se věnuji některým implementačním detailům.

Hlavními přínosy této práce tedy bude vytvoření cenově příznivé platformy pro sledování a ladění CAN komunikace. Dále pak možnost použít tyto navržné SW moduly na již vyvinuté a v budoucnu vyvíjené projekty a eventuálně budoucí tvorba samostatného komerčního produktu.

2 HW

Jedním z důvodů pro volbu sběrnice ethernet oproti USB je snadná přenositelnost výsledné aplikace. Jak serverová tak klientská část, která má formu webové stránky, jsou na jednom místě. Zařízení je tak možno diagnostikovat bez nutnosti dalšího obslužného SW, odpadají tak problémy s kompatibilitou ovladačů pro USB v různých operačních systémech.

Dalším důvodem této volby je vyšší odolnost sběrnice ethernet proti EMC rušení, které v předpokládaném rozsahu aplikací, tedy diagnostika trakčních zařízení, není zanedbatelné. Mezi další výhody sběrnice ethernet patří větší volnost ve fyzickém umístění samostatného zařízení, neboť propojení po síti může být vedeno až do vzdálenosti 500 m bez použití zesilovačů signálu (opakovačů), na rozdíl od sběrnice USB, kde je tato vzdálenost 5 m. Dochází tak k další úspoře nákladů, byť v tomto případě se nejedná o náklady výrobce.

2.1 Současné produkty

2.1.1 National Instruments

NI USB-8473 a NI USB-8472 jsou low-cost produkty pro záznam CAN. Stejně jako pro ostatní produkty této firmy je největší výhoda v možnosti programování v prostředí LabVIEW, které tvorbu aplikací velmi usnadňuje a urchyluje, na druhou stranu samotný vývojový SW nepatří mezi nejlevnější.

Mezi vlastnosti těchto zařízení patří:

- podpora obou verzí identifikátorů
- podpora rychlosti do 125 kbit/s (NI USB-8472) a 1 Mbit/s
- rozlišení času do 1 μs
- napájení 5V
- 1 sběrnice CAN

Cena těchto produktů se pohybuje okolo 9 000 Kč. [22]

2.1.2 Kvaser

Většina produktů firmy Kvaser poskytuje přístup ke dvěma sběrnicím CAN. Výrobce využívá vlastní programovací jazyk Kvaser t pro tvorbu aplikací. HW vlastnosti zařízení jsou srovnatelné s předchozími produkty

Cena produktů se pohybuje okolo 15 000 Kč. [19]

2.1.3 ADFweb

Na rozdíl od předchozích produktů umožňuje většina produktů montáže na DIN lištu, jsou tedy připraveny pro umístění v provozu bez dalších mechanických úprav. Mezi produkty patří například převodníky CAN/ethernet, CANopen/ethernet či záznamník CAN [1]. Kombinace těchto vlastností v jednom produktu ovšem není k dispozici.

2.1.4 Další produkty

Zmíněné firmy nabízí další produkty určené pro CAN, liší se zejména různými rozhraními pro připojení k PC, ať už se jedná o rozhraní PCI, PXI či PCMCIA. Tato rozhraní však nejsou v případě stand-alone zařízení použitelná.

2.1.5 Závěr

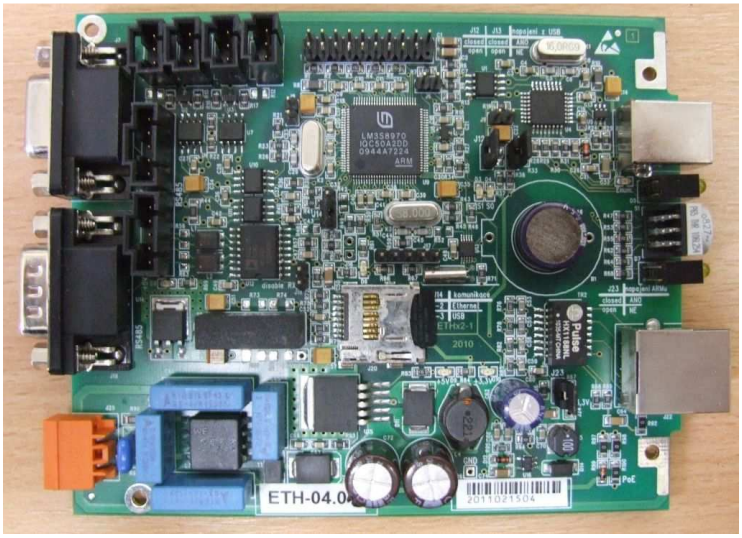
Ze současně dostupných produktů se mi nepodařilo vybrat takový, který by splňoval všechny požadované body. Nejčastějším problémem je absence sběrnice ethernet. V případě kombinace několika prvků, například ze sortimentu ADFweb, je pak jasnou nevýhodou vyšší pořizovací cena této realizace.

2.2 Popis HW platformy POLL

Součástí této práce není návrh výsledného HW, ten mi byl poskytnut společností Poll s.r.o pro vývoj tohoto SW zařízení. HW jednotka ETH-04 je navržena jako komunikační modul a převodník mezi sběrnicemi CAN, ETH, RS-485 a USB. Jádrem platformy je procesor LM3S8970 [20] firmy Texas Instruments. Mezi hlavní rysy tohoto procesoru patří:

- 32-bitová ARM® Cortex™ architektura optimalizovaná pro malé embedded aplikace
- 256 kB flash paměti
- 64kB SRAM
- až 46 GPIO pinů v závislosti na konfiguraci
- dva 16bitové čítače
- tři moduly CAN s podporou CAN verze 2.0A a 2.0B
- řadič ethernetu 10/100 Mbps
- UART, I2C, SSI
- ladění pomocí JTAGu

Jednotku je možno napájet z napětí o jmenovité hodnotě 5-32 V DC, díky čemuž může být trvale umístěna ve vozidle a napájena z palubní sítě. Jako alternativní zdroj napájení je možno použít napájení z PC přes USB, což je využitelné při různých krátkodobých diagnostických zásazích. Výsledná realizace zařízení je zachycena na následujících fotografiích.



Obr. 2.1 DPS

Obr. 2.2 Týl zařízení

Obr. 2.3 Čelo zařízení

3 Analýza zadání

Cílem této práce je navrhnout SW vybavení, realizující diagnostiku a záznam dění na sběrnici CAN. Práci je možné rozdělit na několik částí. Jednou z nich je návrh aplikace běžící v zařízení. Zařízení může být například trvale umístěno v palubní síti kolejových vozidel nebo jej lze připojit pouze dočasně při krátkodobých analýzách. Druhou částí je webová aplikace, přes kterou se s zařízením komunikuje a ovládá se jeho činnost.

Aplikace na straně serveru tedy zajišťuje vlastní zpracování CAN zpráv, ukládání zpracovaných dat a jejich posílání. Minimální požadavky na server jsou následující:

- Podpora verzí protokolu CAN 2.0A a CAN 2.0B s možností přepnout mezi jednotlivými verzemi.
- Možnost přepínání rychlosti komunikace.
- Záznam příchozích zpráv s rozlišením času v milisekundách
- Komunikace s obslužným PC

Na straně klienta je třeba implementovat minimálně následující funkčnosti:

- Zobrazení dat poslaných serverem
- Ovládání aplikace na straně serveru
- Přístup k souborům na SD kartě a práce s nimi (zobrazení, založení, mazání, stažení)

3.1 Webové technologie

Mezi požadavky pro volbu technologie, kterou bude implementována strana klienta, patří zejména HW nenáročnost a tedy schopnost spustit webové rozhraní i na relativně starších strojích, které mohou být k dispozici při různých servisních zásazích. Na straně klienta je možno vybrat si z několika technologií nebo jejich kombinací, cílem této kapitoly je dát dohromady stručný přehled používaných technologií a vybrat některé vhodné pro tuto aplikaci.

3.1.1 Internetové aplikace založené na Flash Player

Flash Player je prostředek umožňující běh interaktivních aplikací, který začal jako přímočarý nástroj pro přidávání animovaných grafik, ale postupnou evolucí dospěl až do platformy pro vývoj interaktivního obsahu. Na rozdíl od standardu HTML má mnohem širší možnosti v tvorbě uživatelských rozhraní.

Pro komunikace se serverem je možno využít dva způsoby komunikace - stavovou a nestavovou. Nestavová komunikace je realizována pomocí protokolu ***HTTP*** a opět se jedná o klasické paradigma dotaz-odpověď, kdy po odpovědi je spojení uzavřeno. Stavovou komunikaci realizuje například ***XML Socket Server***, což bývá aplikace napsána v některém z vyšších jazyků (C#, Java), která s klientem komunikuje na bázi dokumentů ***XML***. Na rozdíl od nestavové komunikace udržuje otevřené spojení, dokud jej jeden z účastníků nezruší.

3.1.2 AJAX

Asynchronnous Javascript And XML není jedna technologie, spíše je to označení pro soubor technik. ***AJAX*** se zabývá komunikací pouze ze strany klienta. Základním stavebním kamenem ***AJAX***u je komunikační objekt ***XMLHttpRequest***, který umožňuje asynchronní komunikaci se serverem. V případě synchronní komunikace (otázka - odpověď) dojde při předání

požadavku k vyčkání na odpověď, v tento okamžik tedy aplikace nereaguje na jakékoliv požadavky.

V případě asynchronní komunikace aplikace udržuje otevřené ***HTTP*** spojení a pokračuje dál ve své činnosti. Data ze serveru zpracuje až v okamžiku, kdy jsou připravena a odeslána, této metodě se říká ***AJAX-Push*** (někdy též Reverse AJAX či Comet). V obou zmíněných případech komunikace je možno odpověď od serveru pomocí JavaScriptových funkcí zpracovat a včlenit do dokumentu, tedy odpadá nutnost tvorby stránky na serveru, jejího stažení a tím překreslení původního obsahu [14].

Tato forma předávání informací skýtá určité nedostatky, nebo spíše odlišnosti, od způsobu, na který jsou uživatelé prohlížečů v současné době zvyklí. Jedná se například o vizuální rozlišení nově příchozích dat, nefunkční tlačítko zpět, respektive nefunkčnost dle očekávání, nemožnost přidat aktuální stav stránky do oblíbených položek a podobné související s nejčastějším statickým formátem stránek. Dalším „problémem" AJAXu je nemožnost tzv. Cross-site scripting, tedy nemožnost jednoduchým způsobem odkazovat do jiné domény, než ze které skript pochází. Pro některé z těchto nedostatků již existují řešení, například tzv. Yellow fading technika řešící nová data, nebo různé forma „uchovávání stavu" stránky v rámci adresy. Na druhou stranu se v rámci této aplikace není třeba většinou těchto omezení zabývat [2].

Ačkoliv je přímo v názvu uvedeno právě XML, tak odpověď serveru ve formátu XML být vůbec nemusí, je možno posílat jak prostý text, tak jiné formáty, například JSON.

3.1.3 Websocket

Jednou z novějších technologií, která umožňuje obousměrnou komunikaci mezi serverem a klientem je ***Websocket.*** Jedná se o technologii vzdáleně vycházející z ***AJAX***u, která nabízí jednoduché rozhraní obsahující tři

události. První z nich je událost Onopen, která je volaná při otevření spojení a může sloužit jako indikátor, že je již možno posílat zprávy. Druhá událost Onclose obdobně oznamuje uzavření spojení. Poslední událost Onmessage, je volána ve chvíli, kdy přijde zpráva. Tělo zprávy je předáno ve vlastnosti *data* dané metody a jedná se o prostý text. Příklady použití jsou uvedeney například v [13] a [18].

V době psaní této práce však technologie ***WebSocket*** není nejpoužívanějšími verzemi prohlížečů (Firefox 4, Internet Explorer 8, Opera 11) plně podporována, zejména kvůli bezpečnostním problémům, jak je demonstrováno v [3]. Širší podpora na straně prohlížečů se očekává s podporou HTML5 [11].

3.1.4 Server-Sent Events

Další poměrně novou technologií jsou ***Server-Sent Events***. Tato technologie definuje interface, který umožňuje otevřít spojení schopné přijímat zprávy pomocí metody push. To tedy serveru umožňuje, aby zprávy posílal „sám od sebe“ v okamžiku, kdy má data připravena, na rozdíl od neustálého dotazování ze strany klienta (polling technologie) Na rozdíl od ***WebSocketu*** komunikují ***Server-Sent Events*** čistě pomocí protokolu ***HTTP***.

Plný popis této technologie je k nalezení v [12], příklad implementace pak v [17]. Vzhledem k tomu, že se jedná o součást HTML5, tak stejně jako v předchozím případě, nejsou ***Server-Sent Events*** příliš podporovány současnými prohlížeči (plně GoogleChrome, částečně pak Opera), rozšíření se opět plánuje právě s podporou HTML5.

3.1.5 JSON

JavaScript Object Notation také není webovou technologií v pravém slova smyslu, nýbrž se jedná o odlehčený formát pro výměnu dat. Je jednoduše čitelný i zapisovatelný člověkem a snadno analyzovatelný i

generovatelný strojově. Je založen na podmnožině programovacího jazyka JavaScript. JSON využívá dva typy struktur:

- Kolekce párů název:hodnota – podle jazyku realizovaná jako objekt, record, asociativní pole atd...
- seřazený seznam hodnot – podle jazyku realizovaný nejčastěji jako pole, vektor, sekvence, atd...

Hodnotou je textový řetězec (v uvozovkách), číslo, logická hodnota true či false, hodnota null, objekt nebo pole, tímto způsobem je tedy možné struktury vnořovat. Více informací o formátu JSON je možno nalézt v [16] .

3.1.6 Silverlight

Jedná se o relativně nový nástroj, jehož první verze byla uveřejněna v roce 2008. Microsoft Silverlight je další alternativou a přímou konkurencí Flashe. Silverlight se však může pochlubit několika architektonickými funkcemi, které Flash nemá - nejvýznamnější je skutečnost, že je založený na zredukované verzi .NET, a umožňuje proto vývojářům psát kód u klienta v čistém C#. Stejně jako v případě Flashe je pro zapojení do prohlížeče použitá technika plug-inu. Pro aplikace Silverlightu platí obvykle obdobné restrikce jako pro obyčejné webové stránky. Například, aplikaci Silverlightu je dovoleno vytvářet soubory a přistupovat k souborům, platí pro ně ale obdobná omezení jako pro cookies z ostatních webových stránek.

Vzhledem k tomu, že programování pro Silverlight prakticky znamená programování v C#, je možnost dosáhnout bohatšího uživatelského rozhraní, do kterého lze zapracovat i přehrávání médií ve formátu WMA, WMV či MP3. Další informace o produktu Silverlight je možno nalézt v [15] a na oficiálních stránkách [21].

3.1.7 Závěr

V předchozích kapitolách byly rámcově nastíněny možnosti současných technologií pro obousměrnou komunikaci klient-server. V následující tabulce jsou shrnuta jednotlivá pro a proti uvedených technologií.

Technologie	Pro	Proti
AJAX	Jednoduchost použití, možnost Ajax push	Javascript, užší možnosti uživatelského rozhraní
Flash	Bohaté uživatelské rozhraní	Plug-in, velikost výsledných aplikací, horší práce s textem a tabulkami
Silverlight	Bohaté uživatelské rozhraní, C#	Plug-in, velikost výsledných aplikací
Server-Sent Events	Push model	HTML5
WebSocket	Jednoduché API	HTML5, bezpečnostní problémy

Tab. 3.1 Pro a proti webových technologií

U Javascriptu je možné, že bude u klienta vypnutý, což se ovšem dá detekovat a upozornit uživatele, že pro plnou funkčnost je potřeba mít JS povolen. V případě plug-inů pro Flash či Silverlight je možné, že na klientských strojích nebudou nainstalovány, což je bez přístupu k internetu neodstranitelná chyba. Navíc jsou možné komplikace s různými verzemi plug-inů a prohlížečů.

Vzhledem k tomu, že původní záměr této práce je možnost spustit diagnostiku na co nejširším spektru prohlížečů, je standard HTML5 zařazen v kategorii proti, protože v současné době není příliš podporován a čeká se na jeho budoucí rozšíření.

Co se týče uživatelského rozhraní, nemusí být graficky nijak náročné, proto by pravděpodobně možnosti dané technologiemi Flash a Silverlight nebyly plně využity, užší možnosti u JavaScriptu jsou způsobeny omezenými

vlastnostmi standardu HTML4 a je tedy možné, že výsledný dojem z grafického uživatelského rozhraní by mohl být lepší.

Na základě těchto údajů jsem se rozhodl klientskou část aplikaci implementovat pomocí technologie AJAX. Serverovou část budu vzhledem k omezeným možnostem procesoru implementovat v jazyce C.

3.2 CAN

Následující kapitola pojednává o sběrnici CAN a aplikační vrstvě CANopen, jsou zde probrány některé CANopen protokoly a požadavky na aplikační rozhraní pro práci s CAN sběrnicí.

3.2.1 CANopen

Jedná se o protokol vyšší vrstvy založený na sběrnici CAN. Jedná se o standardizovaný protokol nejen pro embedded zařízení s flexibilními konfiguračními možnostmi. Standardy CANopen jsou vydávány a zastřešovány organizací CiA [4].

3.2.1.1 Rejstřík objektů CANopen

V rejstříku objektů jsou pro dané zařízení definována všechna data, která se používají ke komunikaci po CANopen. K identifikaci každého objektu v rejstříku slouží 16bitový index a 8bitový subindex. Rozdělení indexů pro jednotlivé typy shrnuje následující tabulka

Index	Data
0000	Nepoužíváno
0001 - 001F	Statické datové typy
0020 - 003F	Komplexní datové typy
0040 - 005F	Komplexní datové typy výrobce
0060 - 007F	Statické datové typy pro dané zařízení
0080 – 009F	Komplexní datové typy pro dané zařízení
00A0 - 0FFF	Rezervováno pro budoucí použití
1000 - 1FFF	Komunikační data
2000 - 5FFF	Data specifická podle výrobce
6000 - 9FFF	Data specifická podle typu zařízení
A000 – BFFF	Standardní interface
CFFF – FFFF	Rezervováno pro budoucí použití

Tab. 3.2 Rejstřík objektů CANopen

3.2.1.2 Objekt SYNC

Synchronizační objekt slouží jako základní hodiny na síti. Je realizován modelem producent – spotřebitel s jednou nepotvrzovanou službou, která nepřenáší žádná data. Objekt SYNC má identifikátor s velmi vysokou prioritu (0x1005).

3.2.1.3 Time Stamp (TIME)

Objekt TIME slouží k přenosu aktuálního času po sběrnici. Je realizován modelem producent – spotřebitel s jednou nepotvrzovanou službou, která přenáší 6 bytů dat typu TIME_OF_DAY, což je struktura obsahující počet ms od půlnoci daného dne a počet dní od 1.1.1984.

3.2.1.4 PDO protokol

Process Data Object je komunikační protokol určený pro předávání krátkých zpráv s vysokou prioritou – nejčastěji se jedná o real-time data. Maximální délka těchto zpráv je 8 bytů. Zprávy jsou vysílány broadcastem a

periodicky. V případě, že je vysílán synchronizační objekt SYNC, umožňuje PDO protokol vysílání synchronních zpráv, jinak je přenos asynchronní.

3.2.1.5 SDO protokol

Service Data Object protokol umožňuje přístup k libovolným datům z rejstříku objektů. Na rozdíl od PDO, tento protokol vytváří spojení mezi dvěma uzly v síti a umožňuje poslat libovolné množství dat. Pro spojení pomocí SDO protokolu je potřeba nakonfigurovat příslušné kanály na straně klienta a serveru (vlastníka rejstříku objektů).

3.2.1.6 NMT protokol

Protokol Network Management řídí slave zařízení v síti. Skrz tento protokol je možno jednotlivé uzly řídit a uvést do jednotlivých stavů inicializovaný, předoperační, operační a zastavený. Podle toho ve kterém stavu se zařízení nachází komunikuje buď pouze SDO, nebo SDO i PDO protokolem (stav operační).

3.2.2 Nastavení CAN sběrnice

V mnoha případech stačí zařízení pouze připojit, jednorázově nakonfigurovat, sesbírat potřebná data, odpojit a analyzovat. V případě, že zařízení má provádět dlouhodobější záznam je potřeba myslet například na situace, kdy dojde například k výpadku napájecího napětí. S touto situací se pojí nutnost načtení předchozího nastavení, protože zařízení, které by bylo pokaždé potřeba znovu nakonfigurovat, pravděpodobně nebude patřit k úspěšným a oblíbeným. Dalším problémem souvisejícím s dlouhodobými záznamy je omezená kapacita záznamového média

V případě tohoto zařízení je pro úspěšné připojení k sběrnici CAN nakonfigurovat následující proměnné:

- Číslo sběrnice, které odpovídá fyzickému rozhraní připojenému k procesoru

- Požadovaná komunikační rychlost, která by měla být zvolena z možností daných standardem
- Verze komunikačního protokolu 2.0A s normálním nebo 2.0B s rozšířeným identifikátorem

Pro uchování těchto údajů existují dvě možnosti. Za prvé je to konfigurace uložená „natvrdo“ v programu, to má výhodu při poruše nebo absenci záznamového média. Na druhou stranu jasnou nevýhodou tohoto řešení je nemožnost přenést takto používané zařízení do prostředí s jinými parametry bez nutnosti zásahu do zdrojového kódu. Druhou možností je již zmíněné záznamové médium, kde po přenesení stačí zásah do jednoho konfiguračního souboru.

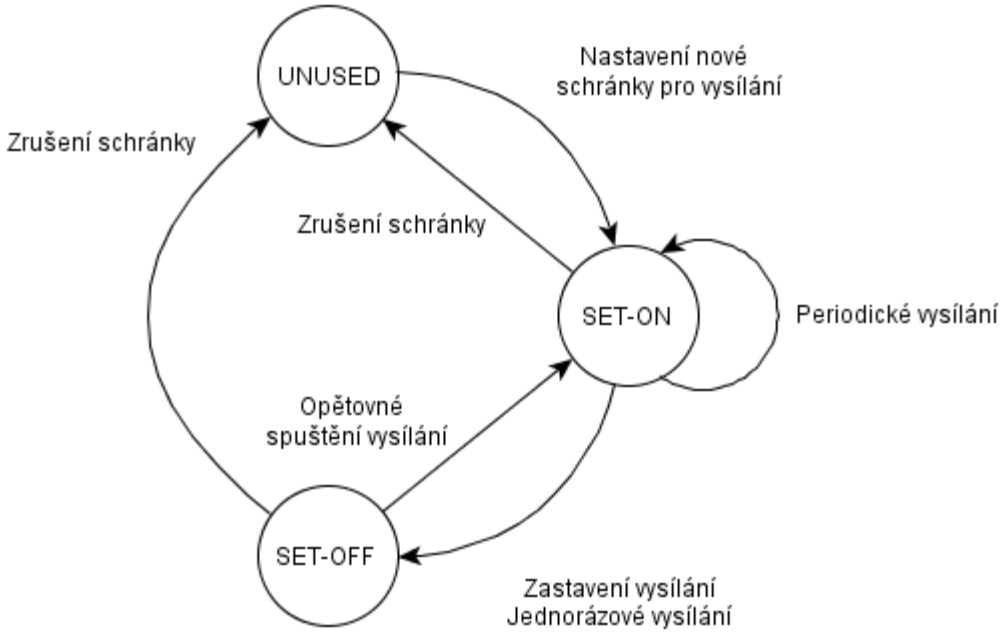
3.2.3 Vysílání vlastních zpráv po CAN

Procesor LM3S9870 obsahuje tři CAN moduly, přičemž každý z těchto modulů může využívat až 32 objektů označovaných jako schránky. Každou z těchto schránek lze nastavit pro příjem či vysílání různých identifikátorů popřípadě určitého rozsahu identifikátorů (maska identifikátoru).

Pro obsluhu jednoho modulu (další jsou identické) je tedy potřebné zajistit následující funkčnosti:

- vyčtení současného nastavení schránek
- nastavení schránky pro příjem/vysílání
- získání prázdné schránky, pokud taková existuje, pro nastavení nového vysílání
- získání schránky podle určitého identifikátoru, pro změnu současného nastavení
- odeslání dat na sběrnici
- zastavení vysílání z určité schránky
- uvolnění nepoužívané schránky

Z výše uvedeného plyne, že každá schránka určená pro vysílání může být popsána stavovým automatem se třemi stavy, mezi kterými přechází na základě uživatelských požadavků a konfigurace vysílání dle následujícího obrázku



Obr. 3.1 Stavový automat schránek CAN

V současném návrhu je potlačena možnost odesílat jednorázové zprávy se zpožděním od zadání. Pokud je schránka nastavena na jednorázové vyslání, přejde do stavu SET-OFF, tento stav tedy uchovává nastavení předchozího identifikátoru a případných dat, což umožňuje vysílání opětovně spustit, bez nutnosti opětovné konfigurace, což se ukázalo jako jedna z chyb u prvního návrhu. Konfiguraci schránek je, stejně jako v předchozím případě konfiguraci sběrnice, možno uložit do souboru, z kterého se po restartu obnoví.

3.2.4 Příchozí zprávy po sběrnici CAN

Jak bylo řečeno v předchozí kapitole, schránka může být nakonfigurována buď na vysílání nebo na příjem, přičemž jedna schránka může, přijímat více zpráv, pokud je lze popsat identifikátorem a maskou udávající, které bity z identifikátoru mají být použity, respektive udává, které bity jsou při průchodu hw filtrem modulu ignorovány a nezáleží na jejich hodnotě.

Stejně jako v předešlém případě je možno konfiguraci uložit do souboru, ale navíc se zde jeví jako opodstatněná metoda konfigurace několika schránek na příjem všech identifikátorů přímo ve zdrojovém kódu programu. Vzhledem k tomu, že se má jednat o záznamové zařízení, je vysoce pravděpodobné, že počet schránek nastavených na příjem zpráv bude výrazně vyšší než počet schránek nastavených na vysílání.

3.3 Práce se soubory

3.3.1 Záznamové médium

Záznamové médium slouží k ukládání příchozích dat, stejně jako je možné jej využít pro uložení konfiguračních dat pro aplikaci.

Největší část vyhrazeného prostoru je samozřejmě vyhrazena pro ukládání zpráv na sběrnici CAN. U každého záznamového zařízení je třeba specifikovat formát a množství ukládaných dat, u formátu je ještě možnost rozhodnout se mezi zápisem čitelným člověkem nebo strojově. Strojově čitelný formát má výhodu úspory místa, na druhou stranu je v tomto případě potřeba počítat s možností přístupu k datům i jiným způsobem než přes webové rozhraní (například vyjmutím karty a použitím obyčejné čtečky), je tedy vhodné implementovat jednoduchý převod do čitelného textu, například formátu csv.

Jedním ze způsobů záznamu by také bylo zaznamenávat, dění na jedné CAN lince vždy do jednoho souboru, což by mělo výhodu v úspoře místa,

alespoň 1 bit (v případě dvou sběrnic) u každé zprávy, na druhou stranu by se tím zkomplikovala možnost porovnání a vzájemného působení na různých sběrnicích. Toto řešení by se tedy dalo aplikovat v případě nastavení pouze jedné sběrnice.

Co se rozsahu ukládaných dat týče, je tedy potřeba zaznamenávat: nastavení sběrnice, viz kapitola 3.2.2, to ovšem nemusí být u každé zprávy stačí jednou v každém souboru se záznamem. U každé zprávy je potřeba uchovat:

- číslo sběrnice
- čas zprávy s rozlišením v milisekundách
- identifikátor zprávy
- data zprávy
- počet dat
- směr TX/RX

Mezi další problémy související zejména s dlouhodobým záznamem dat patří omezená kapacita záznamového média, tedy nutnost buď zastavit záznam při plném médiu, nebo průběžné odmazávání starších záznamů. Pro odmazávání starších souborů slouží předpoklad maximálního uloženého počtu zpráv, respektive doba nejstaršího záznamu. Jestliže chyba není tak závažná, aby vyvolala okamžitý (respektive v rámci rozumného časového období) servisní zásah pak nemá cenu ji nadále uchovávat a je účelnější zanechat prostor pro akutálnější informace.

Na rozdíl od ukládaných dat, kde je potřeba šetřit místem, mohou být konfigurační soubory pro jednotlivé části dle předchozích kapitol uloženy jako prostý text, zejména kvůli možnostem editace.

3.3.2 FileSystem v programu

Předchozí kapitola pojednává o ukládání na obecném médiu, může se ovšem stát, že z nějakého důvodu nebudou tato data přístupná, například z důvodu poškození či absence tohoto média. V tomto případě tedy není možné načíst diagnostickou stránku a je velmi komplikované tuto chybu přesně diagnostikovat, protože k výsledku nenačtení dat vede více možností, jako třeba ztráta napájení, porušení komunikačního vodiče a podobně.

Jednou z možností jak upozornit na chybu záznamového média je mít na chybějící je použít souborový systém v datech, který je k dispozici jako součást modulu lwIP [6]. V okamžiku překladu zdrojového kódu je možno díky utilitě makefsdata vytvořit binární podobu například webové stránky a a následně ji připojit k výslednému projektu. Díky tomu by tedy bylo možné načíst alespoň nějakou formu základní diagnostiky s informací o nepřístupné kartě.

Je samozřejmostí, že souborový systém v datech není plnohodnotným souborovým systémem, protože do něj například není možno zapisovat, je také nutno dbát na omezenou paměťovou kapacitu programu.

4 Design

Jak vyplynulo z analýzy, lze aplikaci v mikrokontroleru rozdělit na několik částí.

- Server zpracovávající http požadavky
- Modul zpracovávající zprávy CAN
- Modul obsluhující soubory

V následujících řádcích je o jednotlivých modulech pojednáno do větší hloubky.

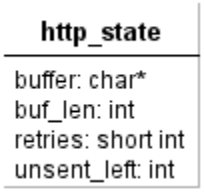
4.1 Server

Modul serveru slouží k rozpoznání příchozího spojení, stará se o vytvoření prostoru pro odpověď v závisloti na úspěchu či neúspěchu zpracování požadavku a odeslání výsledků. Jedná se o rozšíření příkladu poskytnutého s modulem lwip stack dle .

Z možných HTTP metod je implementována pouze metoda GET, na další metody nebyl požadavek, nicméně eventuální implementace by v tomto modulu nebyla obtížná. V případě rozpoznání známého požadavku dojde k jeho delegaci na modul zpracování komunikace. V případě, že požadavek není rozpoznán, je odeslána chybová odpověď.

4.1.1 http_state

Http_state je nosná struktura udržující informace o aktuálním http spojení. Je možno ji znázornit například jako Obr. 4.1



Obr. 4.1 HTTP_state

Jako hlavní prvek zde vystupuje buffer, do kterého se ukládá odpověď vytvořená aplikací a který je po naplnění odeslán klientovi. Vzhledem k tomu, že velikost bufferu se generuje dynamicky v závislosti na volném prostoru, obsahuje struktura ještě údaj o jeho velikosti. Prvek retries obsahuje údaj o počtu opakování spojení, slouží k rozpoznání chyby a eventuálnímu ukončení neúspěšného spojení.

4.2 Modul komunikace

Modul komunikace slouží ke zpracování řetězce předaného serverem. V tomto modulu dochází k parsování a identifikaci prvků podle komunikačního protokolu, naplnění struktur z dat požadavku a zavolání obslužné funkce.

Podle návratové hodnoty jednotlivých funkcí je vytvořena odpověď, kterou je naplněn buffer dle 4.1.1.

4.2.1 Komunikační protokol

Komunikační protokol je seznam zpráv, které si mezi sebou vyměnují klient se serverem. Formát je dán následujícím předpisem:

`http://adresa_serveru/zprava?data1=hodnota1&data2=hodnota2&...~~`.Dvojice znaků `~` slouží k ukončení posledních odeslaných dat. Tímto způsobem je zajištěná možnost posílat data variabilní délky, standardní pevné oddělovače pak zjednodušují parsování řetězce na straně serveru.

Parametr `zprava` nabývá jednu z hodnot daných Tab. 4.1. O počtu, jménech a možných hodnotách dat se zmiňuji v jednotlivých podkapitolách.

Zpráva	Použití zprávy pro
getCANM	Získá zpráv z bufferu
createDir	Vytvoření adresáře
deleteDir	Vymazání adresáře
createFile	Vytvoření souboru pro záznam
deleteFile	Smazání souboru
showDirTree	Načtení adresářové struktury z média
setCANBus	Nastavení sběrnice
setCANM	Nastavení schránky pro vysílání
stopCANM	Zastavení vysílání schránky
deleteCANM	Uvolnění schránky
saveCANM	Uložení nastavení schránek
readCANBusSet	Vyčtení nastavení sběrnic CAN
readCANMSet	Vyčtení nastavení schránek
saveCANBusSet	Uložení nastavení sběrnic

Tab. 4.1 Seznam zpráv

4.2.1.1 getCANM

Zpráva na jejíž žádost server odešle několik posledních zpráv, které se na sběrnicích CAN objevily. Z důvodu úspory přenášených dat jsou zprávy CAN posílány v binárním režimu. V případě, že od poslední žádosti žádné nové zprávy nejsou, posílá chybovou hlášku. Pro tuto zprávu nejsou uvažována žádná uživatelská data.

Data odpovědi (jedna zpráva) mají následující význam:

Data	Délka [bitů]
Číslo sběrnice	3
Směr (TX/RX)	1 (1 - TX, 0 - RX)
Délka dat zprávy	4
Typ identifikátoru(A/B)	1 (1 – B, 0 – A)
Identifikátor	31
Data zprávy	64
Čas	32

Tab. 4.2 Formát CAN zprávy

Touto tabulkou je popsána jedna zpráva CAN, takže tato sekvence se neustále opakuje.

4.2.1.2 createDir

Zpráva na jejíž žádost server na záznamovém médiu vytvoří nový adresář. Data pro tuto zprávu obsahují název adresáře, ten by měl být bez diakritiky, omezení délky názvu je 256 znaků. Jednou z podmínek je dostatek místa na médiu. Odpověď od serveru je interpretována jako text, obsahuje buď název vytvořeného adresáře s prefixem `CDir-` nebo chybovou hlášku, pokud se vytvořit nezdařilo.

4.2.1.3 deleteDir

Zpráva na jejíž žádost server na záznamovém médiu smaže adresář. Mazaný adresář nemusí být prázdný. Data pro tuto zprávu obsahují název adresáře, ten by měl být bez diakritiky, omezení délky názvu je 256 znaků. Odpověď od serveru je interpretována jako text, obsahuje buď název smazáného adresáře s prefixem `DDir-` nebo chybovou hlášku, pokud se smazat nezdařilo (nelze smazat soubor pokud se do něj současně zapisuje...).

4.2.1.4 createFile

Zpráva na jejíž žádost server na záznamovém médiu v aktuálním adresáři vytvoří nový soubor pro záznam zpráv. Data pro tuto zprávu obsahují název souboru, ten by měl být bez diakritiky, omezení délky názvu

je 256 znaků. Jednou z podmínek je dostatek místa na médiu. Odpověď od serveru je interpretována jako text, obsahuje buď název vytvořeného souboru s prefixem `CFile-` nebo chybovou hlášku, pokud se vytvořit nezdařilo. V případě úspěšného vytvoření je nový záznam hned přesměrován do tohoto souboru.

4.2.1.5 deleteFile

Zpráva na jejíž žádost server na záznamovém médiu smaže soubor. Data pro tuto zprávu obsahují název souboru, který má být smazán, omezení délky názvu je 256 znaků. Odpověď od serveru je interpretována jako text, obsahuje buď název smazaného souboru s prefixem `DFile-` nebo chybovou hlášku, pokud se soubor smazat nezdařilo, například protože je momentálně používán.

4.2.1.6 showDirTree

Zpráva na jejíž žádost dojde k vyčtení stromové struktury adresářů uložených na médiu. Pro tuto zprávu nejsou žádná uživatelská data. Odpověď od serveru je interpretována jako text formátovaný jako JSON a použití ilustruje následující příklad odpovědi a podle ní vygenerovaný strom na Obr. 3.1.

```
   {"Konfigurace":[["sbernice.txt","schranky.txt"]],"WEB":[
{"CSS":["default.css"],"HTML":["index.html"],"JS":["basic.js
"]}],"Zaznamy":[{"dir1":["log20111220141516.dat","log2011122
1101510.dat"]}]}
```

Obr. 4.2 Stromová struktura

4.2.1.7 setCANBus

Zpráva na jejíž žádost dojde k nastavení sběrnice CAN. Daty pro tutou zprávu jsou číslo sběrnice, požadovaná komunikační rychlost a verze protokolu CAN 2.0A či 2.0B. Verzi protokolu CAN je možno vynechat, v tom případě je nastavena jako CAN 2.0B. V případě úspěšného nastavení vrací server data nového nastavení. V případě neúspěchu vrací chybovou hlášku s příčinou.

4.2.1.8 setCANM

Zpráva na jejíž žádost dojde k nastavení schránky pro vysílání zpráv CAN. Daty pro tuto zprávu jsou číslo sběrnice, na kterou se má vysílat, id CAN zprávy, data zprávy a perioda vysílání. V případě úspěšného nastavení vrací server data nového nastavení. V případě neúspěchu vrací příčinu. Pokud je perioda nulová zpráva je vyslána jen jednorázově.

4.2.1.9 stopCANM

Zastaví vysílání CAN zprávy s danou ID. Data pro tuto zprávu obsahují identifikátor CAN zprávy, která má být zastavena. Odpověď od serveru obsahuje původní ID s prefixem `sID-`, pokud se podařilo vysílání vypnout, nebo chybovou hlášku.

4.2.1.10 deleteCANM

Uvolní dříve nastavenou schránku. Data pro tuto zprávu obsahují identifikátor CAN zprávy, která má být uvolněna. Odpověď od serveru obsahuje původní ID s prefixem `dID-`, pokud se podařilo vysílání zprávy vypnout a uvolnit schránku, nebo chybovou hlášku.

4.2.1.11 saveCANM

Zpráva na jejíž žádost dojde k uložení současného nastavení schránek na záznamové médium. Data pro tuto zprávu jsou číslo sběrnice, identifikátor zprávy, data zprávy, perioda a příznak zda jde o remote zprávu. Vrací chybovou hlášku, pokud se do souboru nepodařilo zapsat (plné médium). Přepíše původní soubor s konfigurací.

4.2.1.12 readCANBusSet

Zpráva na jejíž žádost dojde k vyčtení nastavení CAN sběrnic. Pro tuto zprávu nejsou uživatelská data. Odpovědí serveru je textový řetězec obsahující konfigurace sběrnic podle kapitoly 3.2.2.

Příklad: `b0s250000b|b1s500000a` udává, že sběrnice s označením 0 je nakonfigurována na rychlost 250 kbit/s a podporuje rozšířený identifikátor, zatímco sběrnice 1 je nakonfigurována na rychlost 500 kbit/s a rozšířený identifikátor nepodporuje.

Oddělovač	Význam
b_n	Číslo sběrnice
S	Komunikační rychlost sběrnice
a, b	Verze protokolu
|	Oddělení konfiguračních bloků

Tab. 4.3 Formát nastavení sběrnice

4.2.1.13 readCANMset

Zpráva na jejíž žádost dojde k vyčtení současného nastavení schránek pro CAN. Pro tuto zprávu nejsou uživatelská data. Odpovědí serveru je textový řetězec obsahující současné nastavení schránek. Odpověď má následující význam

Oddělovač	Význam
b_n	Číslo sběrnice
i	Identifikátor
l	Délka identifikátoru
m	Data zprávy
p	Perioda zprávy
r, n	r-remote n-normální
s, a	s- nevysílá a- vysílá
|	Oddělení konfiguračních bloků

Tab. 4.4 Formát nastavení CAN schránek

Příklad:

```
b0|i503l2maabbp0rs|i603l411223344p1000na...|b1|i603...
```

Znamená, že na sběrnici 0 je nastavená schránka, která jedenkrát vyslala zprávu s identifikátorem 0x503 a daty `aa bb`, momentálně je neaktivní. Další nastavenou je schránka s identifikátorem 0x603 a daty `11 22 33 44`, která momentálně vysílá s periodou 1000 ms.

4.2.1.14 Chybové hlášky

Chybové hlášky jsou tři znaky dlouhé textové řetězce, které server odesílá v případě, že se nepodaří kladně vyhodnotit požadovanou zprávu, přehled chybových kódů shrnuje Tab. 4.5

Kód chyby	Význam
000	Nejsou nové zprávy na CAN
001	Není volné místo
002	Nesprávný název
003	Adresář obsahuje používaný soubor
004	Špatné číslo sběrnice
005	Není nastavena schránka s daným ID
006	Nelze smazat otevřený soubor

Tab. 4.5 Chybové kódy

4.3 Modul CAN

SW část realizující ovladač HW modulu CAN procesoru. Pro každou fyzicky realizovanou sběrnici existuje jedna instance tohoto ovladače. Tento ovladač dále zajišťuje nastavení schránek HW modulu procesoru pro příjem či vysílání a zpracování příchozích zpráv. Příchozí zprávy se zpracovávájí v přerušení, vzhledem k tomu, že tato operace musí být krátká, je u schránky, která data přijala nastaven bit, který indikuje její obsazení. V aplikační smyčce jsou pak takto označené schránky vybrány.

5 Implementace

5.1 Použité technologie

- Eclipse Ganymede 3.4.1.
- C++ a překladač CGT C2000 5.2.1

Při implementaci v jazycích C a C++ mi velmi pomáhaly knihy [9], [10] a zejména [5], zabývající se nejrůznějšími aspekty implementace C++. Největší přínos této knihy pro mě byl v kapitole o spojování částí projektu v C a C++.

5.2 FreeRTOS

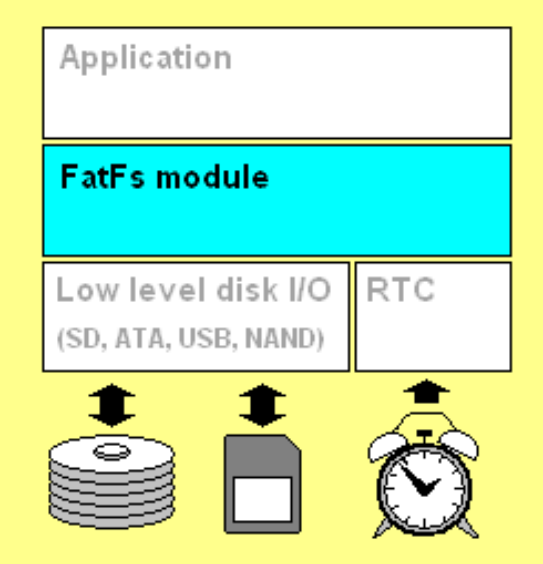
Systémová část je založena na jádru FreeRTOS, který se ve firmě Poll s.r.o. používá již několik let. FreeRTOS je real-timový OS pro embedded zařízení, který podporuje mnoho rozličných architektur. FreeRTOS je uvolněn pod licencí GPL, z níž je vyňat vlastní aplikační kód. Jádro je navrženo jako malé a jednoduché, skládá se z několika souborů v jazyce C. V assembleru jsou implementovány pouze některé, většinou na architektuře závislé, funkce. Operační systém umožňuje rozpad aplikace do několika nezávislých úloh (tasks) z nichž každá má vlastní zásobník, případně coroutine, které sdílejí jeden zásobník dohromady, což snižuje nároky na RAM. FreeRTOS byl v minulosti zvolen zejména pro:

- Testovaný a osvědčený systém, což eliminovalo část komplikací při jeho zavádění
- S tím spojené snížení nákladů na vývojové práce
- Možný přechod na operační systémy OpenRTOS a zejména SafeRTOS, který díky své certifikované bezpečnostní úrovni SIL3 může být zákazníky požadován v budoucích aplikacích

Pro více informací o operační systému viz [8].

5.3 Modul souborového systému

Knihovna FatFs (Generic FAT File System Module) je modul nezávislý na aplikační a na I/O vrstvě použitého HW. Jeho postavení nejlépe vystihuje autorův obrázek Obr. 5.1.



Obr. 5.1 Modul souborového systému

Knihovna FatFs byla zvolena zejména pro:

- Nezávislost na HW
- FAT kompatibilní s OS Windows
- Malá velikost přeloženého kódu
- Možnosti konfigurace ovlivňující funkčnost
- Dřívější zkušenosti s touto knihovnou

Mezi možnožnosti konfigurace knihovny patří použití knihovny pro read-only filesystem a 4 úrovně minimalizace, které postupně zamezují použití funkcí jako je třeba změna atributů adresáře, otevření adresáře či vyhledávání v souborech.

Na dané jednotce ETH-04 probíhá komunikace se záznamovým médiem, micro-SD kartou, po sběrnici SPI, pro vlastní funkci knihovny je tedy nutné implementovat přes tuto sběrnici následující potřebné funkce:

- disk_initialize – která připraví médium pro čtení/ zápis
- disk_status – stav disku neinicializován/nepřítomen/chráněn proti zápisu
- disk_read – přečte sektor z média
- disk_write – zapíše sektor na médium
- disk_ioctl – řídí další funkce a specifické rysy daného záznamového média
- get_fattime – získá údaj z hodin reálného času

Čas pro knihovnu FatFs je ukládán jako 32bitové slovo s rozlišením 2 s. V případě, že aplikace nevyužívá RTC, je nutné vrátit nenulovou hodnotu [7].

5.4 Metoda main

Metoda main po konfiguraci hw, inicializaci FreeRTOSu a jednotlivých modulů vytvoří vlákno FreeRTOSu. Toto vlákno je periodicky spouštěno a realizuje tak hlavní aplikační smyčku, která se stará o aktualizaci RTC a volá smyčku CAN, která zpracovává zprávy uložené ve schránkách. Zbytek aplikace funguje na přerušeních.

5.5 File Manager

File Manager je třída, která řídí přístup k souborům. Zapouzdřuje potřebné operace pro oba typy filesystémů, které se v aplikaci používají, tedy pro knihovnu FatFs (kapitola 5.3) a programový filesystem (kapitola 3.3.2). Samotná jejich implementace je řešena v nižších modulech. Za zmínku stojí operace otevření soboru, priorotně se aplikace snaží nalézt soubor na záznamovém médiu, pouze v případě, že se toto nezdaří, hledá se soubor stejného jména v programu. Momentálně je možné mít otevřen pouze jeden

soubor ke čtení, bez ohledu na to, z kterého systému a jeden soubor pro zápis na paměťové médium.

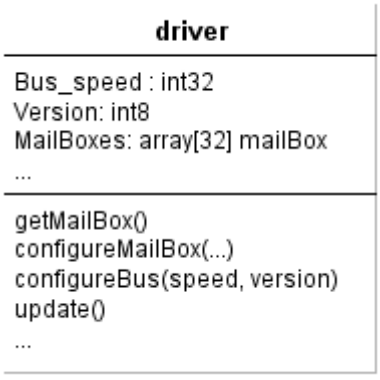
Při každém restartu zařízení je založen nový soubor pro záznam zpráv. Soubory pro záznam také vznikají každou hodinu, porovnáním času vzniku a aktuálního času, nebo po překročení limitního počtu záznamů v souboru. Všechny automaticky generované soubory mají název založený na hodinách reálného času, jedná se o prefix „log“ následovaný dobou vytvoření tohoto souboru. Uživatelské soubory mohou mít v podstatě libovolný název.

5.6 Modul CAN

Modul CAN realizuje ovládání sběrnic a nastavování jednotlivých schránek. Pro každou sběrnici existuje jedna instance třídy, jejímiž atributy jsou příslušné nastavení a pole schránek CAN.

5.6.1 canDriver

Třída implementující ovládání sběrnice. Pro každou fyzickou sběrnici existuje jedna instance, při jejím vzniku je naplněno pole MailBoxes ukazateli na instance třídy mailBox, která realizuje jednu schránku.



Obr. 5.2 Třída canDriver

5.6.1.1 getMailBox

Metoda, která vrací ukazatel na prázdnou schránku, pokud bylo možné ji získat, jinak NULL. Volá se v jen při vzniku instance ovladače.

5.6.1.2 configureBus

Konfigurace sběrnice zadanou rychlostí a verzí identifikátoru. Verze identifikátoru může být vynechána, pokud se tak stane, je automaticky doplněna jako 2.0B.

5.6.1.3 configureMailBoxSend

Nakonfiguruje schránku pro vysílání. Metoda postupně prochází celé pole schránek a hledá, zda je již daný identifikátor používán. Současně je ukládán i index schránky ve stavu `UNUSED`. Pokud je v konfiguraci schránka, jejíž ID se již používá, je nastavení této schránky přepsáno novými daty. Pokud taková schránka neexistuje, a existuje `UNUSED` schránka, je změněn její stav. V případě změny nastavení je nová zpráva ihned vyslána na sběrnici. V této implementaci se pole schránek plní odzadu. Není-li volná schránka vrací se chyba `NO_FREE_MAILBOX`, jinak `OK`.

5.6.1.4 configureMailBoxRec

Nakonfiguruje schránku pro přijímání zpráv. V současné implementaci se tato metoda volá pouze po restartu, kdy je 24 z celkových 32 schránek pro sběrnici nastaveno na příjem všech identifikátorů.

5.6.1.5 freeMailbox

Metoda postupně prochází pole schránek a hledá schránku nakonfigurovanou na vysílání daného identifikátoru, pokud ji nalezne, nastaví ji do stavu `UNUSED`. Tím je tato schránka volná pro opětovné pozdější použití. Pokud ji nenajde, nic se neděje. Vrací `OK`.

5.6.2 mailBox

Metoda update schránky u schránky zkontroluje její stav, přičemž u schránek ve stavu `UNUSED` a `SET_OFF` nevykonává žádnou činnost. U schránek ve stavu `SET_ON` se porovná „čas“ naposledy odeslané zprávy + perioda odesílání oproti „aktuálnímu času“. Ovšem vzhledem k tomu, že hodiny reálného času se aktualizují s poměrně dlouhou periodou, je pro

porovnání využito tiků procesoru. Pokud je aktuální čas větší, dojde odeslání obsahu schránky a zaznamenání času odeslání.

5.7 Klient

5.7.1 Obecně

Implementace klientské stránky aplikace spočívá v tvorbě webových stránek a obslužných JavaScriptů pro obsluhu jejích událostí a zpracování dat od serveru. Vzhledem k tomu, že jsou mezi prohlížeči odlišnosti, co se týče implementace jednotlivých funkčností a existuje spousta vzájemných drobných nekompatibilit, je v první verzi rozchozena funkčnost pouze pro prohlíže FireFox (v době psaní této práce byla aktuální stabilní verze FF4). Tím není naplněn prvotní záměr podporovat co nejširší množinu webových prohlížečů, ale při budoucích úpravách je možné tuto podporu dále rozšiřovat. Pro implementaci

5.7.2 Komunikační modul v JS

Komunikační modul je javascriptový objekt, který zprostředkovává spojení se serverem, obsahuje metody pro spojení, odeslání dat a zavolání funkce, která zpracuje navrácená data. Základní a nejdůležitější public funkcí je `getResponse(request, dataToSend, callBack, method, binary)`, kde jednotlivé parametry jsou:

- request – jedna ze zpráv podle navrženého komunikačního protokolu
- dataToSend – naformátovaná data, která se posílají s výše uvedenou zprávou
- callBack – odkaz na funkci, která má zpracovat data navrácená serverem
- method – http metoda, kterou je požadavek zaslán, prozatím pouze GET, další metody možná v budoucnu

- binary – parametr určující, zda je odpověď zpracovávána jako plain text nebo jako binární posloupnost

Dalšími, privátními, funkcemi jsou zejména různé kontrolní mechanismy, zajišťující zejména kontrolu formátování odesílaných dat a vlastní vytvoření objektu XMLHttpRequest.

5.7.3 User-Interface

Uživatelské rozhraní má formu webových stránek, každá z nich využívá funkce JavaScript, které z reakcí na události vytváří zprávy podle navrženého komunikačního protokolu popsaného v kapitole 4.2.1. V této kapitole se nachází informace o zpracování některých odpovědí a jejich začlenění do vlastní diagnostické stránky.

5.7.3.1 Aktuální provoz

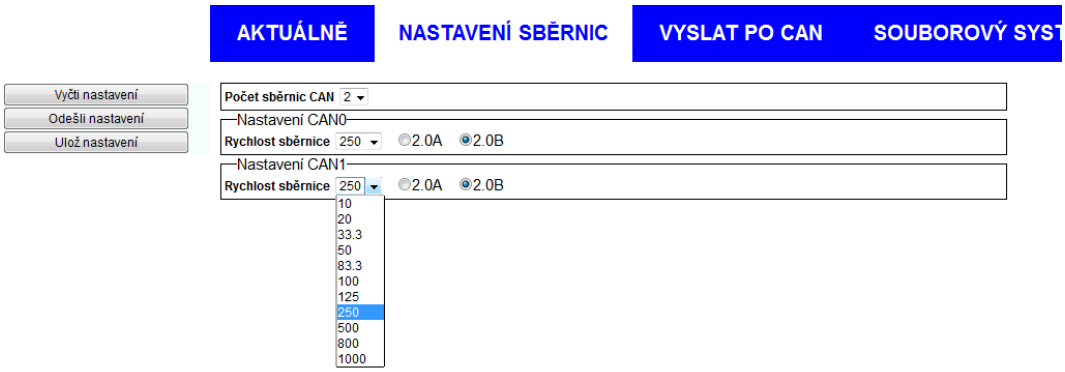
Tento diagnostický list zobrazují reálné dění na sběrnici. Nejdůležitější zprávou je tedy getCANM. Odpověď na tuto zprávu je po příchodu rozdělena na bloky pevné délky podle Tab. 4.2 Každý z těchto bloků je dále zpracován funkcí JavaScriptu, pomocí které postupně jsou získány jednotlivé údaje z bloku.Tyto údaje jsou následně použity k tvorbě nového řádku tabulky s posledními zprávami, aktualizaci celkového počtu zpráv a aktualizaci celkového počtu zpráv.

Identifikátor je použit k vytvoření třídy ve smyslu standardu HTML, všechny řádky se stejným identifikátorem pak mají stejnou třídu. Pomocí nich je pak zajištěna například filtrace, kdy se pomocí CSS atributu display všechny tyto řádky skryjí. Maximum zobrazovaných řádků v tabulce je 100 000 záznamů, pak dojde k odmazávání nejstarších.

5.7.3.2 Nastavení sběrnic

Pro nastavení sběrnic je vytvořena následující stránka Obr. 5.3. Po jejím načtení (událost `onLoad`) je serveru odeslána zpráva `readCANBusSet`, z které

je vygenerována následující tabulka. Stejného efektu lze docílit kliknutím na tlačítko `Vyčti nastavení`.



Obr. 5.3 Nastavení sběrnic

V případě konfigurace je třeba z rolovacího seznamu zvolit počet sběrnic a dále rychlost(uváděna v kbit/s) a verzi identifikátoru pro každou z nich. Tlačítko `Odešli nastavení` pak serveru odešle zprávu `setCANBus`, čímž dojde ke změně aktuálního nastavení. Tlačítkem `Ulož nastavení` dojde k odeslání zprávy `saveCANBus` a tím k přepsání původního konfiguračního souboru.

5.7.3.3 Nastavení schránek

Pro nastavování CAN zpráv pro vysílání je vytvořena následující diagnostická stránka. Při jejím načtení dojde k odeslání zprávy `readCANMset` a z odpovědi podle Tab. 4.4 je pak vygenerována tabulka jako například na obrázku, podobného efektu je docíleno při kliku na tlačítko `Současné nastavení`, přičemž původní obsah tabulky je smazán.

Obr. 5.4 Nastavení zpráv CAN

Stavy jednotlivých schránek (vysílá/nevysílá), jsou znázorněny graficky, bílé podbarvení nastaveného řádku, značí aktivní schránku, bleděmodré zastavenou. Poslední barevnou indikací je červená, která značí chybu při odesílání nastavení (např snaha odeslat data mimo hex rozsah či zápornou periodu). Klepnutí na tlačítko `Nastavení nové zprávy` přidá nový řádek, do něhož je poté možno nastavit žádané hodnoty.

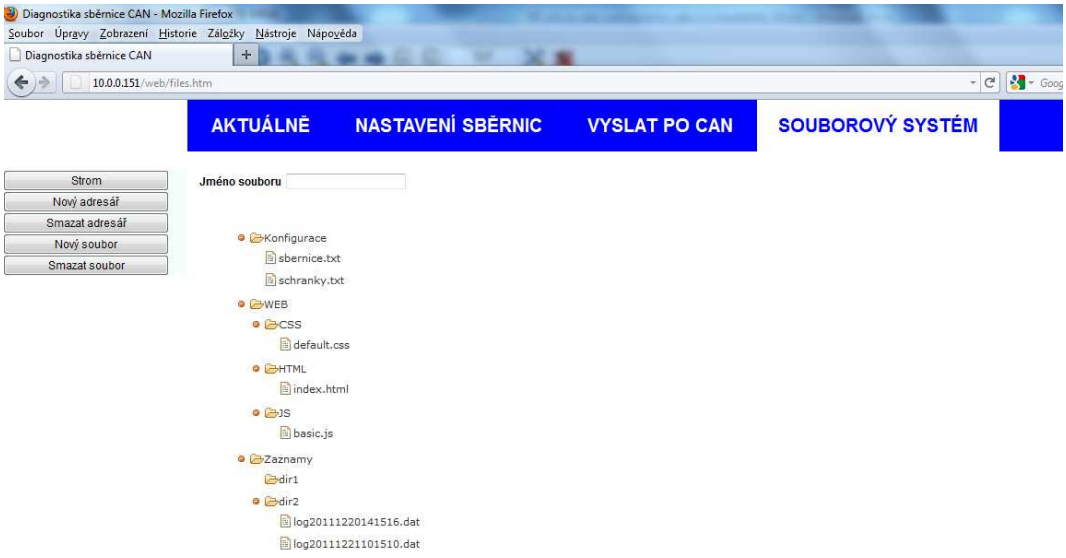
Tlačítkem ▶ pak proběhne kontrola, zda nejsou data mimo povolený rozsah, a pokud tomu tak není, jsou odeslána do zařízení. Následně pak začne vysílání příslušné zprávy. Perioda 0 znamená jednorázové vyslání zprávy. Tlačítkem ■ se zpráva s příslušným ID přestane vysílat, schránka tak přejde do stavu `SET_OFF`. Poslední tlačítko ✕ vymaže tento řádek a dojde ke zrušení schránky s tímto ID, tedy přechodu do stavu `UNUSED`.

Tlačítko `Ulož nastavení` odešle jednotlivé řádky tabulky nastavení ve zprávě `saveCANM` podle 4.2.1.11, na straně serveru jsou pak data zapsána do konfiguračního souboru. Původní nastavení konfiguračního souboru je tímto krokem přepsáno.

Zbylé ovládací prvky slouží taktéž pro celou tabulku a povolí vysílání, zastaví vysílání nebo uvolní všechny zobrazené schránky.

5.7.3.4 Souborový systém

Obsluha souborového systému momentálně není kompletní, zatím je možno pouze vyčíst adresářovou strukturu, která je uložena na SD-kartě. Z dalších funkčností je zde možnost založení a mazání adresářů a založení či mazání souborů. Do nově založeného souboru je poté přesměrován záznam zpráv. Největší dosud nevyřešený problém této části spočívá v nemožnosti stáhnout z karty více souborů na jedno kliknutí. Toto omezení je implementováno v prohlížečích z bezpečnostních důvodů, kdy každý požadavek na download musí být uživatelem potvrzen. V praxi na výkonných serverech je toto omezení obcházeno metodou komprese všech požadovaných souborů do jednoho archivu, ten je pak uživatelem potvrzen a stažen. Na straně tohoto serveru však takovéto možnosti nejsou, proto tato možnost nepřipadá v úvahu. Pro hromadné stahování souborů je možno využít standardního prokolu FTP.



Obr. 5.5 Ovládání souborů

5.7.3.5 Chybové hlášky

Vzhledem k tomu, že všechny navržené chybové hlášky dle Tab. 4.5 mají v textové formě délku tři znaky, probíhá jejich detekce pouze porovnáním délky odpovědi. Pokud je délka odpovědi 3, je v souboru chyby. js nalezen příslušný text, resp. číselný kód a podle něj je nalezena příčina chyby a vrácen text, pro zobrazení v prohlížeči.

Obr. 5.6 Zobrazení chybové hlášky

6 Závěr

Po provedení rešerše webových technologií a první analýze požadavků na SW, jsem se rozhodl pro použití technologií AJAX, HTML a CSS pro implementaci klientské části aplikace. První navržené a implementované řešení uživatelského rozhraní je vidět na obrázcích v kapitole 5.7. Serverová část aplikace je naprogramována v kombinaci jazyků C a C++. V obou částech aplikace jsem použil objektový model a navrhl si moduly realizující jednotlivé funkčnosti jako například komunikační protokol přes rozhraní ethernet. Na poskytnutém HW pak došlo k praktickému testování, kdy se mi podařilo zaznamenávat zprávy na sběrnici CAN.

V následujícím období po konzultaci s nově příchozími kolegy dojde k další iteraci sesbírání požadavků na funkčnosti, jejich analýzy a případné implementaci do systému. Vzhledem k modulárnímu charakteru aplikace tyto další kroky nevyžadují hrubé zásahy do jádra a jejich zavedení by nemělo být časově náročné. Dále budu pracovat na vylepšení uživatelského rozhraní, aby reflektovalo změny ve funkčnostech a zároveň působilo přívětivěji s širšími možnostmi uživatelské konfigurace.

Testování výsledného zařízení zatím proběhlo pouze v laboratorních podmínkách. V těch se návrh osvědčil a momentálně se čeká na nasazení zkušebního vzorku do reálného provozu.

Na základě těchto výsledků a rozšíření funkčností se opět přiblížíme k cíli vytvořit komerční produkt, což v současném stavu není možné.

Literatura

[1] ADFweb. *canopen - datalogger CANOpen* [online], http://www.adfweb.com/home/products/datalogger_canopen.asp?frompg=nav2_16, [cit. 2011-03-12]

[2] ASLESON, R. – SCHUTTA N.T. *Ajax : Vytváříme vysoce interaktivní webové aplikace*. 1. vyd. Brno: Computer Press, 2006. ISBN 80-251-1285-3

[3] BARTH, Adam. *Talking to Yourself for Fun and Profit* [online] http://www.adambarth.com/papers/2011/huang-chen-barth-rescorla-jackson.pdf, [cit. 2011-02-11]

[4] CAN-CiA. [online] http://www.can-cia.org/, [cit. 2011-06-11]

[5] CLINE, Marshall. *C++FAQ* [online] http://www.parashift.com/c++-faq/,[cit. 2011-08-11]

[6] DUNKELS, ADAM. *lwIP- A Lightweight TXP/IP Stack* [online] http://www.sics.se/~adam/lwip/[cit. 2011-08-15]

[7] ELM - *FatFs Generic FAT File System Module* [online] http://elm-chan.org/fsw/ff/00index_e.html [cit. 2011-08-22]

[8] FreeRTOS. [online] http://www.freertos.org [cit. 2011-06-11]

[9] HEROUT, Pavel. *Učebnice jazyka C*. 6. vyd České Budějovice: KOPP 2010. ISBN 978-80-7232-383-8

[10] HEROUT, Pavel. *Učebnice jazyka C. 2. díl* 4. vyd České Budějovice: KOPP 2010. ISBN 978-80-7232-367-8

[11] HICKSON, Ian. *HTML5* [online] http://dev.w3.org/html5/spec/Overview.html, [cit. 2011-04-05]

[12] HICKSON, Ian. *Server-Sent Events* [online] http://www.w3.org/TR/2011/WD-eventsource-20110208/, [cit. 2011-04-27]

[13] HICKSON, Ian. *The Web Socket API* [online] http://www.w3.org/TR/2009/WD-websockets-20090423/, [cit. 2011-05-27]

[14] HOLZNER, Steve. *Ajax For Dummies,* 1. vyd. Indianopolis: Wiley Publishing, 2006. ISBN-13"

[15] Interval.cz. *Silverlight - co je Silverlight? | Interval.cz* [online] http://interval.cz/clanky/silverlight-co-je-silverlight/, [cit. 2011-04-11]

[16] JSON [online] http://www.json.org, [cit. 2011-05-11]

[17] KLUČKA, Bronislav. WebNT - *(29): Server-sent Events | HTML5 (HTML 5), CSS3 (CSS 3)* [online] http://www.webnt.cz/29-server-sent-events/, [cit. 2011-04-27]

[18] KLUČKA, Bronislav. *WebNT - (8): WebSocket | HTML5 (HTML 5), CSS3 (CSS 3)* [online] http://www.webnt.cz/8-websocket/, [cit. 2011-05-02]

[19] Kvaser. *kvaser.com - Data Logger* [online] http://www.kvaser.com/en/products/can/data-logger.html, [cit. 2011-03-13]

[20] Luminary Micro, *Stellaris®; ARM®; Cortex©;-M-based MCUs - 8000 Series - LM3S8970 - TI.com* [online] http://www.ti.com/product/lm3s8970, [cit. 2011-02-28]

[21] Microsoft. *Microsoft Web | Silverlight* [online] http://www.microsoft.com/cze/web/silverlight/, [cit. 2011-04-16]

[22] National Instruments. *NI In-Vehicle CAN Data Logger System - CAN Measurement, Analysis, and Data Synchronization - National Instruments* [online] http://sine.ni.com/nips/cds/view/p/lang/en/nid/209237, [cit. 2011-03-13]

[23] ZAKAS, Nicholas Z. *JavaScript pro webové vývojáře*. 1. vyd. Brno: Computer Press, 2009. ISBN 978-80-251-2509-0

Seznam zkratek

- CAN – Controller Area Network
- CiA – CAN in Automation
- FatFs – File Allocation Table File System
- GPIO – General Purpose Input Output
- RTC – Real Time Clock
- WMA – Windows Media Audio
- WMV – Windows Media Video
- WCF – Windows Common Foundation